

Bootstrapping bilinear models of robotic sensorimotor cascades

Andrea Censi and Richard M. Murray¹

Abstract We consider the bootstrapping problem, which consists in learning a model of the agent's sensors and actuators starting from zero prior information, and we take the problem of servoing as a cross-modal task to validate the learned models. We study the class of bilinear dynamics sensors, in which the derivative of the observations are a bilinear form of the control commands and the observations themselves. This class of models is simple yet general enough to represent the main phenomena of three representative robotics sensors (field sampler, camera, and range-finder), apparently very different from one another. It also allows a bootstrapping algorithm based on hebbian learning, and that leads to a simple and bioplausible control strategy. The convergence properties of learning and control are demonstrated with extensive simulations and by analytical arguments.

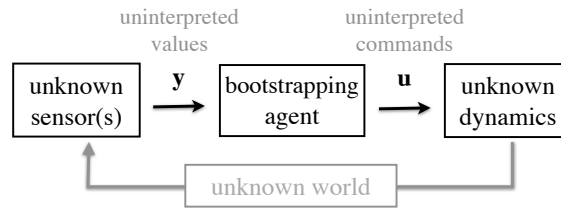


Fig. 1: The bootstrapping problem. Can an agent with no prior information about its sensors and actuators learn a model of its sensorimotor cascade, and to use it for useful tasks? Being able to solve this problem would allow us to realize truly plug-and-play robotics, and would give us insight into some of the profound features of the nature of intelligence.

Control & Dynamical Systems, California Institute of Technology.
{andrea,murray}@cds.caltech.edu

Version 2, Sep 15th: Simulations commented in more detail, notation made more uniform. Expanded and split the discussion of BGDS models to a new technical report.

Version 1, Jul 15th: first version.

1 Introduction

Two features of natural intelligence that we are far from emulating in artificial systems are its adaptiveness and generality. The human neocortex is highly uniform and its parts can be repurposed; for example, the visual cortex is repurposed to process tactile information in blind subjects [4]. Reproducing the same adaptability in artificial systems is a vast problem considered in various forms (and by various names) in several fields, such as AI, machine learning, robotics, and the specialized (and fairly distinct) fields of epigenetic and developmental robotics [1, 16].

A rather extreme, yet concrete, version of the problem has been put forward by Kuipers and colleagues in a long series of papers (see [13, 19, 20] and references therein). Suppose that an agent starts its life with no prior information about its sensors and actuators. It can read the sensors output as a sequence of values, but no semantics is associated to them. Likewise, it does not know how the unlabeled commands it can generate affect the world. The bootstrapping problem concerns creating a model for its sensorimotor cascade from scratch, and using it to achieve useful tasks.

Bootstrapping can be seen as an extreme form of system identification/calibration. Currently, there exist autocalibration techniques that can estimate parametric models of the dynamics (for example, the odometric parameters) or the extrinsic sensor configuration, (although the solutions, rather than general, tend to be tailored to specific sensors or groups of sensors), but always the type of sensors/actuators being calibrated is known a priori. Can a robot learn to use an unknown sensor and unknown actuators? Can the *same* learning algorithm work for a range-finder and a camera? These are, at the moment, open questions. They are important for practical robotics applications: it would be extremely convenient, if you could just attach any sensor to a robot, and the robot would learn how to use it, without tedious programming. More in general, we believe that robotic systems are the perfect benchmark for supposedly “universal learning agents”, which so far have been studied for only perception/classification tasks [6], or as body-less agents [10].

The most complete exemplification of Kuipers and colleagues’ idea of a bootstrapping agent is the Spatial Semantic Hierarchy [12]. They show that it is possible to start from uninterpreted sensor data and build successive layers of representation up to a topological map of the environment. The crucial transition from continuous sensor values to symbolic representations de-

depends on the definition of *trackers*, distinctive features in the sensory stream whose behavior is predicted by the agent’s action. Their work offers several opportunities for extension. One concern is that their results remain largely anecdotal, in the sense that they are illustrated by simulations/experiments, but not by proofs; this makes it hard to build on them. The other concern is that most of the results regard the case of range-finders: while they showed, in principle, that the same design pattern could be applied to different sensor modalities, it is unclear whether exactly the same algorithm could be run unchanged for different sensorimotor cascades. The two aspects of provability and generality are our focus in this paper.

This paper shows that it is possible to design *unsupervised* learning algorithms that learn *generative* models of a robotics sensorimotor cascade, for a wide range of sensors, and use that model to perform useful tasks. Moreover, the model we consider is simple enough that many of its properties can be proved theoretically.

Throughout the paper, we consider three classes of sensors: *cameras*; *range-finders*, devices sensing the distance to the closest obstacle; and *field samplers*, devices that sample a generic spatial field, such as the concentration of a chemical substance. We call these three “canonical” sensors, in the sense that they are representative of many others. For example, the range-finder abstraction encompasses both sparse sonar-like sensors, as well as dense lidar-like sensors. The camera abstraction encompasses RGB and infrared cameras. The field-sampler is general enough to represent olfactory and temperature sensors (see, e.g., [8, 15]). The idea is that, even if these three sensors do not capture all possible sensors, they cover enough ground such that, if we present an algorithm that can work for all of these, then it would be fair to think it is a “generic” algorithm. As for the robot dynamics, we limit the analysis to fully actuated robots controlled in velocity (still, the agent does not know which command is which).

Models are only as good as the actions they generate. To show that the bootstrapping agent has acquired a useful model of its sensorimotor cascade, we consider its performance in the servoing task. Let \mathbf{y} be the observations (for example, the raw pixel intensities returned by a camera), and let \mathbf{u} be the commands. We define servoing as follows.

Problem 1. Given a goal observation \mathbf{y}_* , choose $\mathbf{u}(t)$ such that $\mathbf{y}(t) \rightarrow \mathbf{y}_*$.

This is an interesting task to consider because the problem statement is simple, it makes sense for all sensor modalities, and the ability to solve it means that the agent has captured a significant part of its sensorimotor cascade’s actual model.

Our approach has been to study a model which is fairly general, yet simple enough to be learned and analyzed. In Section 2 we consider the abstract class of *bilinear dynamics sensors* and we design a two-phase bootstrapping strategy. During a first unsupervised learning phase, the agent learns a representation of its sensorimotor cascade using a simple Hebbian-learning

based algorithm. In the second phase, the control action solving the servoing task is given as a function of the learned model. In Section 3 the algorithm is validated in simulation, for various configurations of the three sensors. Section 4 concerns actually proving that the algorithm works for the three canonical sensors.

2 Bootstrapping bilinear dynamics sensors

At the heart of bootstrapping, there is a problem of prediction: how do actions change the agent’s view of the world? Specifically, how do the actions \mathbf{u} change \mathbf{y} ? Any model we decide to use must be general enough to be applied to different sensorimotor cascades, must be informative enough so that we can use it to solve the problem of servoing, and it must be simple enough so that it is easy to learn and analyze. In this section, we argue that the simplest yet useful model for robotics sensorimotor cascades is assuming that $\dot{\mathbf{y}}$ is a bilinear form of \mathbf{y} and \mathbf{u} ; if this holds exactly, we call the sensor a *bilinear dynamics sensor* (BDS). We then study the problem of learning unsupervisedly the model of a BDS and how to use that model for servoing.

Notation and formal definitions

We consider sensors composed of a set of sensory elements (*sensels*) that are physically related to one another. We write the observations as $\mathbf{y} = \{y^s\}_{s \in \mathcal{Y}}$, where s is the sensel position ranging over the *sensel space* \mathcal{Y} . In the case of a camera, the sensels span the visual sphere \mathbb{S}^2 ; s corresponds to a pixel’s direction, and y^s to the intensity measured by that pixel. Real robots have discrete sensors with a finite number of sensels; but to make the derivation simpler we pretend that the sensel space \mathcal{Y} is continuous. The values returned by the sensors lie in a certain *output space* \mathcal{O} . For a color camera, \mathcal{O} would be the RGB space; for a range-finder, \mathcal{O} would be \mathbb{R}^+ (distances). For simplicity, we will just assume \mathcal{O} to be \mathbb{R} ; everything can be extended to more complicated output spaces. At each time, the sensor returns the observations as a function from \mathcal{Y} to \mathbb{R} , assumed differentiable. A formal signature for the observations \mathbf{y} is $\mathbf{y} : \text{time} \rightarrow C^1(\mathcal{Y}; \mathbb{R})$.

We assume that there is an inner product defined on $C^1(\mathcal{Y}; \mathbb{R})$. This means that we have a way to measure the dissimilarity of two observations by the norm induced by the inner product. We use the tensorial notation to represent the inner product. Let y^s represent the value of \mathbf{y} at the sensel $s \in \mathcal{Y}$. We put the index up in “ y^s ” with analogy to covariant tensors. Given two observations \mathbf{y}_1 and \mathbf{y}_2 , their inner product $\langle\langle \mathbf{y}_1, \mathbf{y}_2 \rangle\rangle$ can be represented by contracting y_1^s, y_2^v with a $(0, 2)$ tensor m_{sv} : $\langle\langle \mathbf{y}_1, \mathbf{y}_2 \rangle\rangle = m_{sv} y_1^s y_2^v$. Here, using the Einstein convention, summation (integration) is assumed

over indices that appear twice (up and down). The inner product allows to define a norm $\|\mathbf{y}\|^2 = \langle\langle \mathbf{y}, \mathbf{y} \rangle\rangle$ as well as a conjugation operation $\mathbf{y} \mapsto \mathbf{y}^*$ by $y_s^* = m_{sv} y^v$.

Why using a bilinear model

In the most general case, a continuous-time dynamical system can be written as $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$; $\mathbf{y} = h(\mathbf{x})$, where \mathbf{x} represents the hidden state. However, one seldom sees an explicit model of this kind for sensors such as cameras or range finders, because the function h should encode all information regarding the environment, and a closed form is impossible to write except in the simplest of environments. One alternative representation is focusing on the observations dynamics $\dot{\mathbf{y}} = g(\mathbf{y}, \mathbf{u}, \mathbf{x})$. In Section 4 we show that, for the three canonical sensors, this can actually be written in a closed form. In most cases, the function g depends on the underlying unobservable state \mathbf{x} . An agent that does not have access to the state \mathbf{x} (and its dynamics) cannot learn such a model. This motivates us to look at approximating the observations dynamics by disregarding the dependence on the state, thus looking for models of the form $\dot{\mathbf{y}} = g(\mathbf{y}, \mathbf{u})$. Because the agent has access to \mathbf{y} , $\dot{\mathbf{y}}$, and \mathbf{u} , learning the map g from the data is a well-defined problem.

Rather than trying to learn a generic nonlinear g , which appears to be a daunting task, especially for cases where \mathbf{y} consists of thousands of elements (pixels of a camera), our approach has been to keep simplifying the model until one obtains something tractable. For example, a second-order linearization of g leads to the expression

$$\dot{\mathbf{y}} = \mathbf{a} + A\mathbf{y} + B\mathbf{u} + C(\mathbf{y}, \mathbf{y}) + D(\mathbf{y}, \mathbf{u}) + E(\mathbf{u}, \mathbf{u}). \quad (1)$$

Here A and B are linear operators, but C, D, E are tensors (later we make the tensor notation more precise). If \mathbf{y} and \mathbf{u} have dimensions n and k , then C, D, E have dimensions, respectively, $n \times n \times n$, $n \times n \times k$, and $n \times k \times k$.

We can ignore some terms in (1) by using some assumptions regarding our specific context. For example, if we assume that \mathbf{u} represents a “movement” or “velocity” command, in the sense that if \mathbf{u} is 0, then the pose does not change, and \mathbf{y} does not change as well ($\mathbf{u} = 0 \Rightarrow \dot{\mathbf{y}} = 0$), we can omit the terms \mathbf{a} , $A\mathbf{y}$ and $C(\mathbf{y}, \mathbf{y})$, and we are left with $\dot{\mathbf{y}} = B\mathbf{u} + D(\mathbf{y}, \mathbf{u}) + E(\mathbf{u}, \mathbf{u})$.

If we assume that \mathbf{u} is a symmetric velocity commands, in the sense that applying $+\mathbf{u}$ gives the opposite effect of applying $-\mathbf{u}$, then we can get rid of the $E(\mathbf{u}, \mathbf{u})$ term as well. We are left with the model $\dot{\mathbf{y}} = B\mathbf{u} + D(\mathbf{y}, \mathbf{u})$, where D is a bilinear operator. We can incorporate $B\mathbf{u}$ into the second term by assuming there is a trivial observation whose value is always 1. In conclusion, our *ansatz* for a generic robotic sensor is a bilinear model of the kind $\dot{\mathbf{y}} = D(\mathbf{y}, \mathbf{u})$.

Definition 1. A sensor is a bilinear dynamics sensor (BDS), for a certain choice of control commands \mathbf{u} , if the derivative of \mathbf{y} depends linearly on \mathbf{u} and \mathbf{y} . In formulas, there exists a (1,2) tensor \mathbf{M} such that

$$\dot{\mathbf{y}}^s = \mathbf{M}_{vi}^s \mathbf{y}^v \mathbf{u}^i. \quad (2)$$

Bootstrapping and servoing with BDS

As explained in the introduction, the task we focus is servoing/homing (Problem 1). The solutions we study are two-part strategies, composed by a learning and a control phase. In the first learning phase, the agent builds a representation of its sensorimotor cascade. Then, the agent uses the representation it has built to solve the task during the action phase.

In the learning phase, the agent randomly samples control commands \mathbf{u} from any zero-mean distribution with positive definite covariance. Meanwhile, it estimates three quantities: the average observation at each sense $\bar{\mathbf{y}}$:

$$\bar{\mathbf{y}}^s = \mathbb{E}\{\mathbf{y}^s\}, \quad (3)$$

the (2,0) covariance tensor \mathbf{P} :

$$\mathbf{P}^{sv} = \text{cov}(\mathbf{y}^s, \mathbf{y}^v), \quad (4)$$

and the (3,0) tensor \mathbf{T} defined by

$$\mathbf{T}^{svi} = \mathbb{E}\{(\mathbf{y}^s - \bar{\mathbf{y}}^s) \dot{\mathbf{y}}^v \mathbf{u}^i\}. \quad (5)$$

These expectations can be computed online. For example:

$$\bar{\mathbf{y}}^s(k+1) = \frac{k}{k+1} \bar{\mathbf{y}}^s(k) + \frac{1}{k+1} \mathbf{y}^s(k).$$

We note that these computations can be implemented on a neural architecture; equation (5) is similar to three-way Hebbian learning between \mathbf{y} , $\dot{\mathbf{y}}$, and \mathbf{u} . In fact, the expectation of the product of the three terms $(\mathbf{y}^s - \bar{\mathbf{y}}^s)$, $\dot{\mathbf{y}}^v$, \mathbf{u}^i can be thought as an approximation of the frequency that the three signals are active together.

Lemma 2. Let \mathbf{P} , \mathbf{Q} be the covariance of \mathbf{y} and \mathbf{u} . Then the tensor \mathbf{T} tends asymptotically to:

$$\mathbf{T}^{svi} = \mathbf{M}_{qj}^s \mathbf{P}^{qv} \mathbf{Q}^{ij}. \quad (6)$$

Proof. We can prove that this tends to:

$$\begin{aligned}
\mathbf{T}^{svi} &= \mathbb{E}\{\dot{y}^s (y^v - \bar{y}^v) u^i\} \\
&\quad \text{Substituting the model 2 for } \dot{y}, \text{ and changing indices.} \\
&= \mathbb{E}\{(\mathbf{M}_{qi}^s y^q u^i)(y^v - \bar{y}^v) u^i\} \\
&\quad \text{Independence of } u, y, \text{ and the fact that } \mathbf{M} \text{ is a constant.} \\
&= \mathbf{M}_{qi}^s \mathbb{E}\{y^q (y^v - \bar{y}^v)\} \mathbb{E}\{u^i u^i\} \\
&\quad \mathbb{E}\{y^q (y^v - \bar{y}^v)\} = \text{cov}(y^q, y^v) \\
&= \mathbf{M}_{qi}^s \mathbf{P}^{qv} \mathbf{Q}^{ij}.
\end{aligned} \tag{7}$$

□

In the expression (6), we can observe a general pattern. Every quantity that the agent learns ultimately depends on three factors:

1. *The agent's sensorimotor cascade.* In this case, \mathbf{M} .
2. *The environment statistics.* In this case, the covariance \mathbf{P} represents the effect of the specific environment in which learning takes place. For example, in the case of a camera, the covariance \mathbf{P} depends on the statistics of the environment texture, and it would change in different environments.
3. *The experience the agent had in such environment.* In this case, \mathbf{Q} captures the kind of "training" the agent had in the environment.

In the control phase, the agent uses the learned tensors \mathbf{T} and \mathbf{P} to generate the commands. The following proposition is the main result of this section.

Proposition 3. *Assume that the agent is equipped with a BDS (Definition 1), that it has learned \mathbf{P} and \mathbf{T} using equations (4)–(5), and \mathbf{Q} is positive definite. Then the control law*

$$u^i = -(y^v - y_\star^v)^* \mathbf{T}^{svi} \mathbf{P}_{vq}^{-1} y^q \tag{9}$$

corresponds to a descent direction of the error metric $\|\mathbf{y} - \mathbf{y}_\star\|$. Moreover, if the operators $\{\mathbf{M}_{vi}^s y^v\}_{i=1}^k$ commute, \mathbf{y}_\star is asymptotically stable.

Proof. The first part of the proof shows that the control is a descent direction for $V = \|\mathbf{y} - \mathbf{y}_\star\|^2$. Defining the error signal $e^s = y^s - y_\star^s$, we can write $V = \frac{1}{2} e^s m_{rs} e^r$. The derivative of V can be computed as follows.

$$\begin{aligned}
\dot{V} &= e^s m_{rs} e^r = \dot{y}^s m_{rs} e^r \\
&= [M_{vi}^s u^i y^v] m_{rs} e^r \\
&\quad \text{Expanding } u \text{ using (9).} \\
&= -M_{vi}^s [e^z m_{zw} T^{wx} P_{xp}^{-1} y^p] y^v m_{rs} e^r \\
&\quad \text{Expanding } T \text{ using (6).} \\
&= -M_{vi}^s e^z m_{zw} [M_{qj}^w P^{qx} Q^{ij}] P_{xp}^{-1} y^p y^v m_{rs} e^r \\
&\quad \text{Reordering everything.} \\
&= -y^v M_{vi}^s e^z m_{zw} M_{qj}^w P^{qx} P_{xp}^{-1} y^p Q^{ij} m_{rs} e^r \\
&\quad \text{Tensors } P \text{ and } P^{-1} \text{ cancel.} \\
&= -(y^v M_{vi}^s m_{rs} e^r) Q^{ij} (y^p M_{pj}^w m_{zw} e^z) \\
&\quad \text{Let } g_i = y^v M_{vi}^s m_{rs} e^r. \\
&= -g_i Q^{ij} g_j \leq 0.
\end{aligned}$$

This proves that V never increases. If we prove that g is never 0 in a neighbourhood of \mathbf{y}_\star , then $\dot{V} < 0$, and V is then a Lyapunov function, making \mathbf{y}_\star asymptotically stable. To prove this second part we have to use some tools from nonlinear system theory [18, Chapter 11, page 540]. In general, because the Lie algebra is nilpotent of order 0, any solution of the equation (2) can be written in the form

$$\mathbf{y} = \exp(M_{\cdot 1} b_1) \exp(M_{\cdot 2} b_2) \cdots \exp(M_{\cdot k} b_k) \mathbf{y}_\star, \quad (10)$$

where the b_j are known as the Philip Hall coordinates, and \exp represents the exponential of a linear operator. We are interested only on the behavior near \mathbf{y}_\star , therefore we can linearize each of the term as

$$\exp(M_{\cdot j} b_j) = I + M_{\cdot j} b_j + o(\|\mathbf{b}\|^2).$$

The linearized version of (10) is then

$$\mathbf{y}^s = \mathbf{y}_\star^s + M_{vj}^s b^j y_\star^v + o(\|\mathbf{b}\|^2).$$

From this we get $e^s = M_{vj}^s y_\star^v b^j + o(\|\mathbf{b}\|^2)$, which we substitute in the definition of g_i to obtain

$$\begin{aligned}
g_i &= y_\star^v M_{vi}^s m_{rs} e^r + o(\|\mathbf{b}\|^2) \\
&= y_\star^v M_{vi}^s m_{rs} (M_{qj}^r y_\star^q b^j) + o(\|\mathbf{b}\|^2).
\end{aligned}$$

Recall that we are in a neighborhood of \mathbf{y}_\star (but not precisely at \mathbf{y}_\star). This implies that \mathbf{b} is not zero; otherwise, from (10) we get $\mathbf{y} = \mathbf{y}_\star$. Assuming that the $M_{qj}^r y_\star^q$ commute, we also have that $M_{qj}^r y_\star^q b^j \neq 0^r$. In fact, if they commute, we can write (10) as $\mathbf{y} = \exp(M_{qj}^r y_\star^q b^j) \mathbf{y}_\star$ and the argument of the

exponential must be different from 0. These two together imply that $g_i = y_\star^v M_{vi}^s m_{rs} (M_{ij}^r y_\star^q b^j) \neq 0$ near the origin. Here the abundance of indices is masquerading the simplicity of the assertion. Without indices: suppose there is a vector $v \neq 0$, and a linear operator A such that $Av \neq 0$; then $A^*Av \neq 0$. In this case, $v = b$ and $A_i^v = y_\star^v M_{vi}^s$. \square

Remark 4. It is a classic result [2] that, if a system such as (2) is nonholonomic, there exists no smooth controller that stabilizes y_\star asymptotically. In particular (9) is smooth in y , therefore it cannot work in the nonholonomic case. Instead, the requirement that the operators $\{M_{vi}^s y^v\}_{i=1}^k$ commute is a technical necessity for having a compact proof and can probably be relaxed.

Improvements on the basic strategy

The bootstrapping strategy has a couple of interesting properties: the tensors \mathbf{T} and \mathbf{P} can be learned using simple Hebbian learning, and the resulting control strategy is a bilinear form of the observations y and the error $y - y_\star$. These two properties allow a very efficient engineering implementation, and at the same time make the algorithm implementable using neural networks (“bioplausible”). The only operation that is not bioplausible is computing \mathbf{P}^{-1} . This motivates looking for ways to get around such computation.

Whitening the observations. If the observations had covariance equal to the identity, we could omit the term \mathbf{P}^{-1} . This suggests one way to proceed: find a transformation $z = Wy$ such that z has unit covariance. In the signal processing community, the problem of finding a suitable transformation W is well known and it is called *whitening*. It is a well studied problem because it is needed as a first step before performing independent component analysis [11]. Numerous algorithms exist for whitening, most of them having a neural implementation [5].

Omitting “ \mathbf{P}^{-1} ” from (9). One could also ask whether it is possible to simply omit \mathbf{P}^{-1} even when it is different from the identity. We can prove the following technical condition on \mathbf{P} and \mathbf{M} that makes it possible to omit the \mathbf{P}^{-1} from (9) and still obtain a suitable minimization strategy.

Proposition 5. *Assume that \mathbf{P} and \mathbf{M} commute, in the sense that, defining $\mathbf{P}_v^q = \mathbf{P}^{qr} m_{rv}$, it holds that $\mathbf{M}_{ij}^r \mathbf{P}_i^q = \mathbf{P}_s^s \mathbf{M}_{ij}^s$. Then the control strategy*

$$u^i = -(y^s - y_\star^s)^* \mathbf{T}^{svi} (y^v)^* \quad (11)$$

minimizes the error metric $V = \frac{1}{2} e_s^ \mathbf{P}_v^s e^v$, where $e^s = y^s - y_\star^s$.*

Proof. Consider a definite positive error metric $V = \frac{1}{2} e^r \mathbf{X}_{sr} e^s$ for some definite positive tensor \mathbf{X} . We can compute the gradient flow as follows:

$$\dot{V} = e^r X_{rs} \dot{y}^s = e^r X_{rs} M_{vi}^s y^v u^i.$$

Therefore, any control command of the form

$$u^j = -Q^{ij} e^r X_{rs} M_{vi}^s y^v \quad (12)$$

is a descent direction for V . Write now the control strategy (11) and expand its terms:

$$\begin{aligned} u^j &= -(y_*^v - y^v)^* T^{svj} (y^s)^* \\ &\quad \text{Definition of } *. \\ &= -e^r m_{rs} T^{swj} m_{wv} y^v \\ &\quad \text{Using Lemma 2.} \\ &= -Q^{ij} e^r m_{rs} M_{qi}^s P^{qw} m_{wv} y^v \\ &\quad \text{Definition of } \mathbf{P}. \\ &= -Q^{ij} e^r m_{rs} M_{qi}^s P_v^q y^v \\ &\quad \text{Commutation property.} \\ &= -Q^{ij} e^r m_{rs} P_q^s M_{vj}^q y^v \\ &\quad \text{Definition of } \mathbf{P}. \\ &= -Q^{ij} e^r (m_{rs} P^{sw} m_{wq}) M_{vj}^q y^v \end{aligned} \quad (13)$$

By comparing (12) and (13) one can see that (12) is a descent direction for the quadratic form $\frac{1}{2} e^r X_{sr} e^s$ with $X_{rs} = m_{rv} P^{vw} m_{ws}$. \square

We shall discuss the meaning of the commutation condition when we get to discussing the actual sensors. We will see that in some cases the covariance \mathbf{P} acts as a smoothing operation, and that the tensor \mathbf{M} is similar to a gradient operation, and we will be able to interpret the condition $\mathbf{MP} = \mathbf{PM}$ in more intuitive terms as “the gradient commutes with smoothing”. Also note that now the error function depends on the covariance \mathbf{P} ; in contrast with the previous proposition, now the environment statistics influence the actions.

3 Simulations and experiments

This section shows that the bootstrapping strategy seems to work for the three canonical sensors considered; the next section will justify theoretically some of these findings.

We simulate a planar omnidirectional robot controlled in velocity. The commands are $\mathbf{u} = (u_1, u_2, u_3) = (v_x, v_y, \omega)$. In our case, simulations are precious because we can try several different sensor configurations. We sim-

ulate a 180deg range finder, an omnidirectional camera, and a field sampler, with the sensels placed on a ring. In the three cases, the observations y are, respectively, the range readings, the luminance readings, and the field intensity. We use a custom simulator.¹ The bootstrapping part consists in placing the robot in a randomly-generated map at random places (simulated as a uniform variable on the subset of $SE(2)$ that does not intersect any obstacle), and simulate the sensor output y , \dot{y} when the robot chooses a random command u (simulated as a Gaussian random variable with spherical covariance).

We found out that, if one uses environment shapes which are too simple, the learned model will pick up the characteristic of the environment. For example, if a robot with a 360deg range-finder is always placed in a room of the same size, say 10m, it will learn that, if the readings in front measure 1m, it is likely that the readings in the back measure 9m — this knowledge is represented implicitly in the estimated covariance matrix, which will report strong negative correlation between readings in front and in the back. We do not want to see these effects, which are not representative of the real world, and to prevent them we simulate random environments composed of randomly sampled polygonal walls. See Fig. 1 for examples of the random environments used. It seems that, as long as there is some variability, the results are largely independent of the details of how the randomness is introduced. These observations motivated the definition of the environment’s *symmetry group* (Definition 12) and how it affects the observation covariance.

For simulating a camera sensor, one should choose a random texture for the surfaces: for example, sample a luminance value independently for each 20cm section of the surface, and smooth the result. Choosing structured inputs such as sinusoids introduces unwanted correlation. This motivated the definition of *monotone environment* (Definition 18).

For the field sampler, we simulated a random distribution of point sources with quadratic decay. We simulated various sensors configurations. For the field-samplers, we simulated sensels disposed either in a ring or a 180deg semiring. There is nothing special about the ring configuration: the method would work with an arbitrary disposition and ordering of the sensels. However, putting the sensels on a regular shape produces results which are easier to interpret. We also simulated “normalized” field-samplers, in which the output of a sensel is divided by the sum of the other sensels, such that the sum is one. This “winner-take-all” mechanism is ubiquitous in biological systems. For the camera, we simulated a 180 and 360deg field of view. Likewise, for the range-finder, we simulated 180 and 360deg field of view, plus 180deg field of view with a foveal sensel disposition (denser in the center). We also tried different extrinsic configuration, with the sensor rototranslated with respect to the robot center.

¹ The Python/C++ source code is available at <http://purl.org/censi/2010/boot>.

Figures 2 onward show the learned tensors. In all figures, blue means negative and red positive; each figure is normalized independently from the others. Subfigures *b–c* show the covariance (\mathbf{P}) and information (\mathbf{P}^{-1}) tensors of the simulated sensors. These tensors are given as a function of the sensels position s, v . In these three simulated agents, where the sensels are disposed on a semicircle, we let s, v be the angle with respect to the robot front. The covariance encodes information on the sensel topology. For the camera and range-finder, the covariance is sparse and local: only sensels that are very close to each other are correlated, and the correlation is a function of the sensels distance. The covariance/information matrix act very similarly to convolution/deconvolution operators.

Subfigures *d–f* show the learned tensor \mathbf{T} . If there are n sensels and 3 commands, then \mathbf{T} is a $n \times n \times 3$ tensor. We show the 3 bidimensional slices $\mathbf{T}^{sv(1)}, \mathbf{T}^{sv(2)}, \mathbf{T}^{sv(3)}$. For example, the slice $\mathbf{T}^{sv(1)}$ describes how the linear velocity v_x is related to $y(s)$ and $\dot{y}(v)$. Subfigures *g–i* show the normalized tensor \mathbf{TP}^{-1} used in the control law (9). While \mathbf{T} depends on the environment (because of the internal dependence on the covariance \mathbf{P} , which depends on the environment statistics), \mathbf{TP}^{-1} cancels out the environmental contribution. Perhaps the results for range-finder and camera are easier to interpret. If one imagines the slices \mathbf{T}^{svi} as linear operators that, applied to \mathbf{y} , give $\dot{\mathbf{y}}$, it is evident that the agent learns local spatial gradients; we shall see this theoretically.

We also tried the learning algorithm with real range-finder data from the Rawseeds project² [3], and we obtained similar results, which convinced us of the validity of the simulations. In this case, rather than static pictures, we give links to videos that show the learning in real time. `video:LaserDisplay`³ shows the observations, which consists in the data from two Sick range-finders spliced together. `video:LaserBDSLerning`⁴ shows the evolution of the tensor \mathbf{T} , which is qualitatively similar to the simulation results. `video:LaserCorr`⁵ shows the evolution of the covariance \mathbf{P} : due to the fact that the robot visits regular, structured environments (e.g., corridors), the variance is different at each sensel, and the correlation is not only a function of the sensels distance (see [7] for analogous conclusions for camera data).

Figures 14 onward show the convergence properties of the control law (9) and the simplified control law (11), in simulation. We are interested in evaluating the radius of convergence. We sample numerous environments and goal configurations; then we compute the control law in a volume around the goal configuration. We show the results as “success maps”: we slice the $q = (x, y, \theta)$ volume at the three planes (x, y) , (x, θ) , (θ, y) , and we count the percentage of times the control law pointed the robot in the right direction,

² Data available at <http://www.rawseeds.org>.

³ <http://purl.org/censi/2010/be#LaserDisplay>

⁴ <http://purl.org/censi/2010/be#LaserBDSLerning>

⁵ <http://purl.org/censi/2010/be#LaserCorr>

meaning that it would have decreased the distance to the goal⁶. Light green means $> 99\%$; see the caption for the other values. For the field sampler and camera (Fig. 14-15), we can see that the control law (9) gives local convergence, while the simplified version (11) has worse performance. For the range-finder (Fig. 16), both give robust convergence.

The rest of the paper is dedicated to proving the results suggested by the simulations.

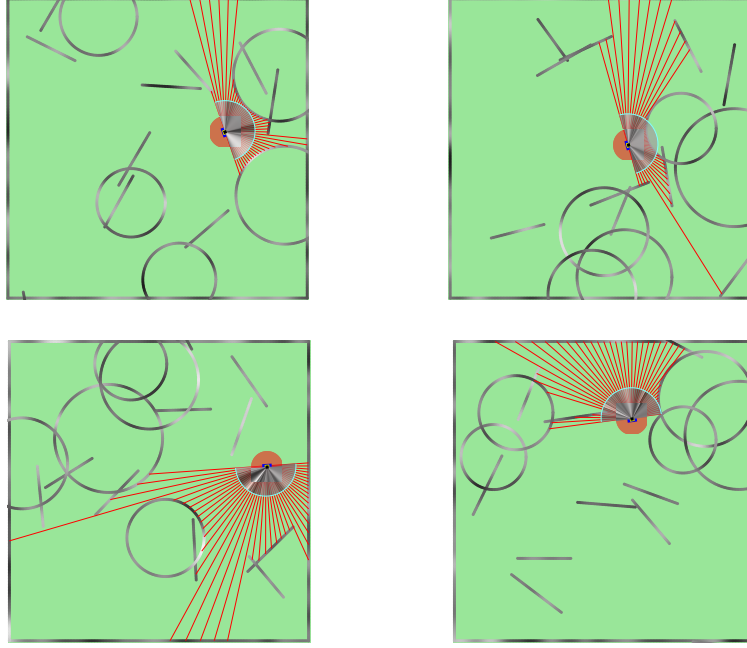


Fig. 1: Examples of random worlds simulated for learning. The red rays represent the range-finder readings. The black and white arc around the robot represent the simulated output of camera. Note the random geometry and the random texture. Figure best viewed on screen zooming in on the details.

⁶ In formulas: let the goal be $\mathbf{q} = (x, y, \theta) = (0, 0, 0)$. Then a “successful” command is one for which $d\|\mathbf{q}\|/dt \propto xu_1 + yu_2 + \theta u_3 < 0$. Note that here we treat $SE(2)$ locally as a subset of \mathbb{R}^3 .

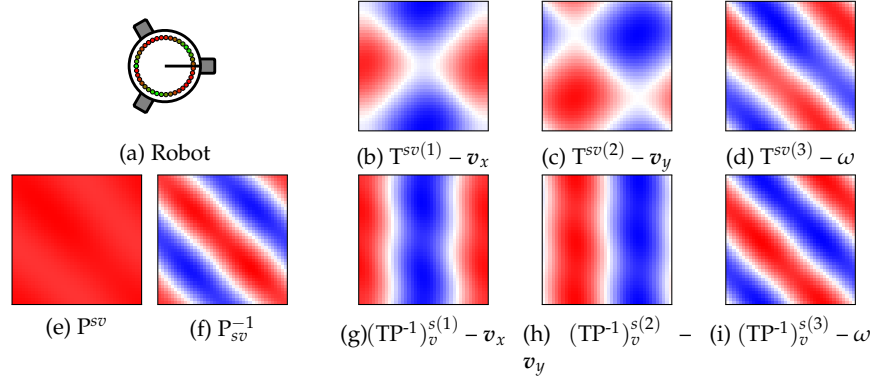


Fig. 2: Tensors learned for robot with a field sampler, with sensels placed on a 360deg ring. Each axis corresponds to an angle on the ring. Red means positive; blue negative; white zero. Subfigures *e-f* show correlation and information matrix, as a function of the sensel positions $s, v \in \mathcal{Y}$, which can be thought of the angle on the sensor ring. The correlation is almost identically 1; this depends on the statistics of the field we simulated. Figure *b-d* show the three slices of the learned tensor \mathbf{T} , for the three commands (v_x, v_y, ω) . The tensor element T^{isv} represents the interaction between the i -th command, the observation y^s and the derivative \dot{y}^v . Figures *g-i* show the 3 slices of the normalized tensor \mathbf{TP}^{-1} . (Images best seen in color.)

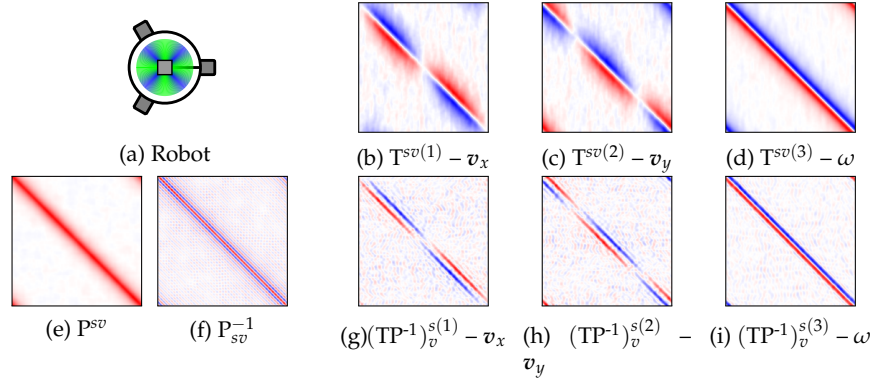


Fig. 3: Tensors learned for robot with omnidirectional camera. See Fig. 2 for a general description of the figures. All pictures are a functions of two sensels $s, v \in \mathcal{Y}$, which corresponds to the pixel orientation. In *e-f*, we can see that the covariance matrix of a camera is sparse and local: only nearby sensels interact. Figures *b-d* show the learned tensor \mathbf{T} ; if one interprets each slice as a linear operator that, applied to \mathbf{y} , gives $\dot{\mathbf{y}}$, it is clear that all three represents functions of the gradient of \mathbf{y} . Depending on the particular environment, the gradient is more or less smoothed by the covariance. This is confirmed theoretically – see Fig. 1. Figures *g-i* represent the slices of the normalized tensor \mathbf{TP}^{-1} : here the effect of the environment statistics are factored away and an even more local operator is obtained.

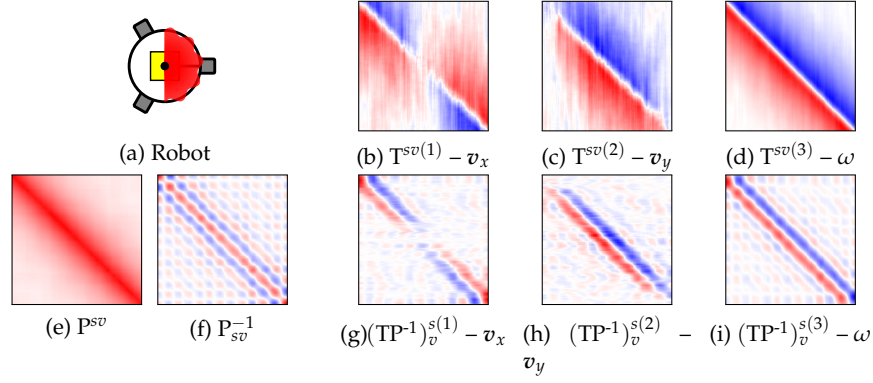


Fig. 4: Tensors learned for robot with range-finder (180deg FOV). See Fig. 2 for a general description of the figures. It is interesting to compare these results with the camera results in Fig. 3. The covariance is less local: this means that, in the environment we simulated, the range readings are more correlated than the covariance; that is, they change less abruptly. As a consequence, the tensor T is less local than the corresponding tensor for the camera. Figures $g-i$ show that the normalized tensor TP^{-1} , where the effect of the environment statistics is removed, has a more local character.

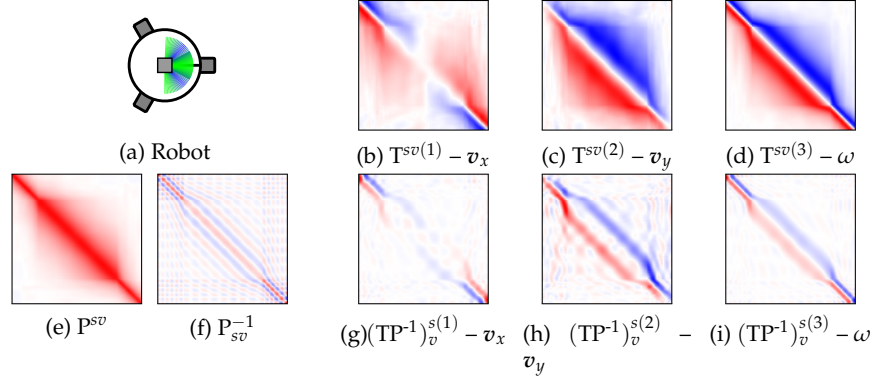


Fig. 5: Tensors learned for a robot with 180deg camera.

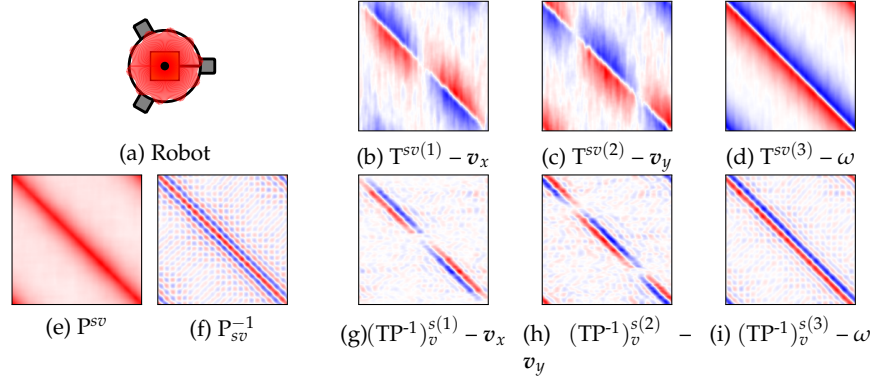


Fig. 6: Tensors learned for a robot with omnidirectional range-finder.

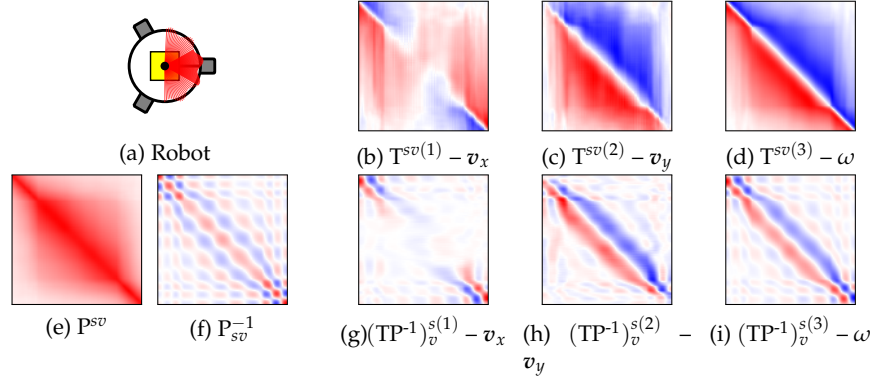


Fig. 7: Tensors learned for a robot with 180deg range-finder, denser in the middle.

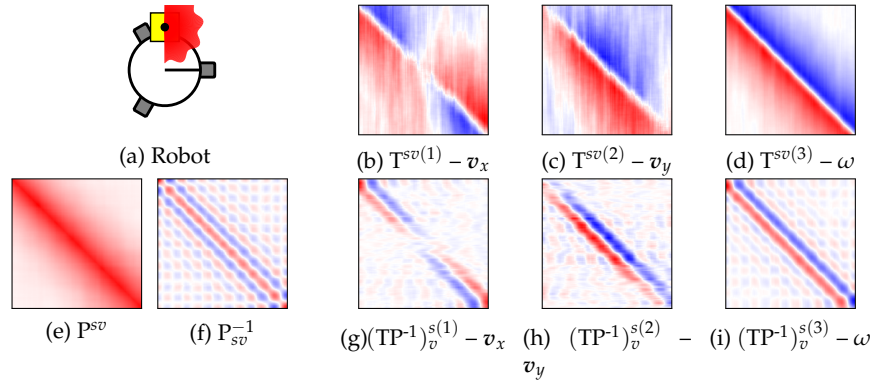


Fig. 8: Tensors learned for a robot with 180deg range-finder, translated laterally.

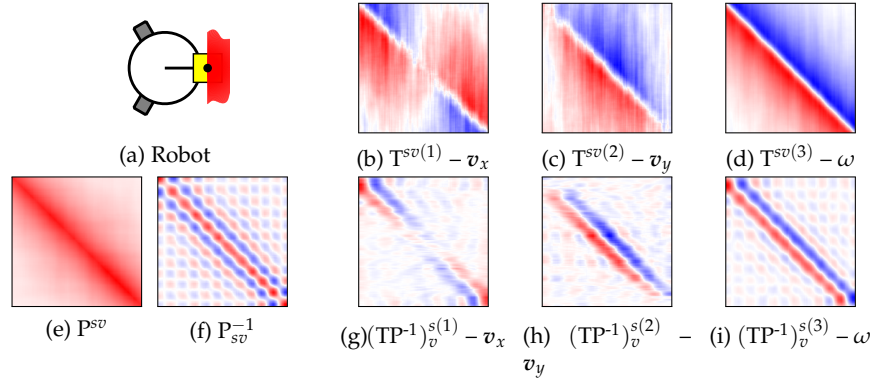


Fig. 9: Tensors learned for a robot with 180deg range-finder, translated forward.

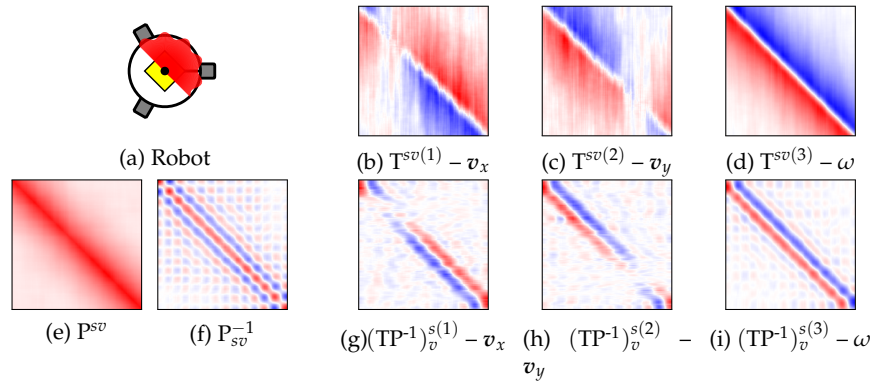


Fig. 10: Tensors learned for a robot with 180deg range-finder, rotated.

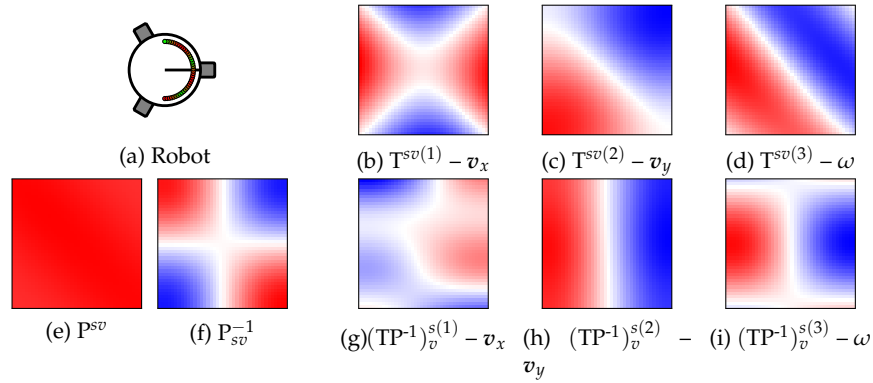


Fig. 11: Tensors learned for a robot with field sampler (sensors placed on a 180deg semiring).

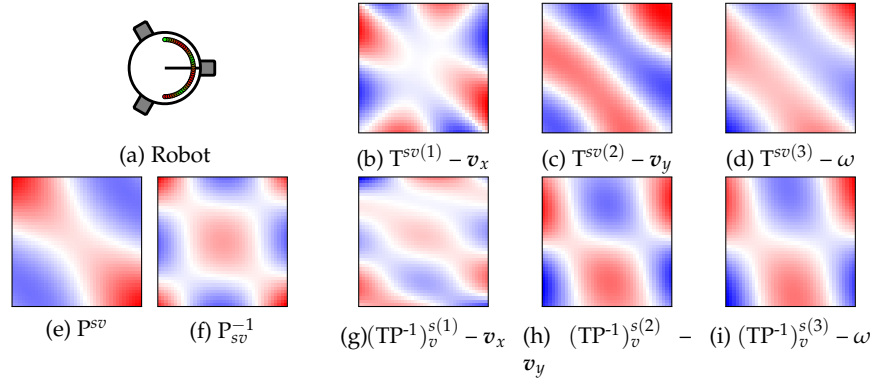


Fig. 12: Tensors learned for a robot with field sampler (sensors placed on a 180deg semiring), with normalization.

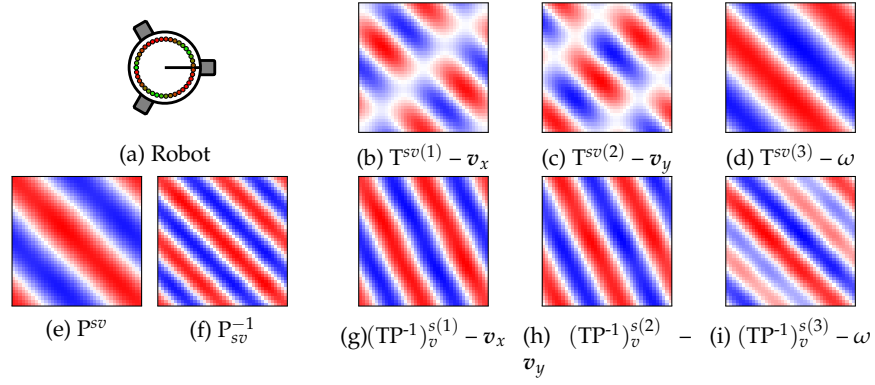


Fig. 13: Tensors learned for a robot with field sampler (sensors placed on a 360deg ring), with normalization.

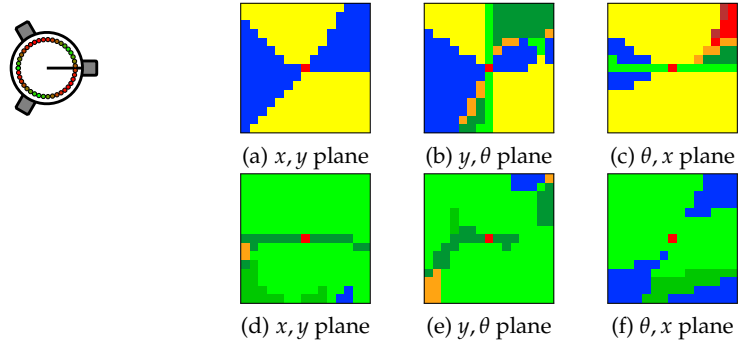


Fig. 14: Statistics of the convergence of the two control laws for robot with a field sampler, with sensels placed on a 360deg ring. Figures (a)-(c) show the results for the simplified control law (11), the figures (d)-(f) show the results for the control law (9). We put the goal at the origin, and considered starting positions sampled in a $1\text{m} \times 1\text{m} \times 45\text{deg}$ parallelepiped around the goal. We show the convergence results along three slices in the planes $x, y, y, \theta, \theta, x$. The figures show the percentage of times (over 200 trials with random environments) that the control law indicated a direction decreasing the error metric. The color scale is: 0% (red) <25% (yellow) >25% (orange) >50% (blue) >75% (dark green) >95% (light green) 100% (bright green). These figures are best seen on a computer screen.

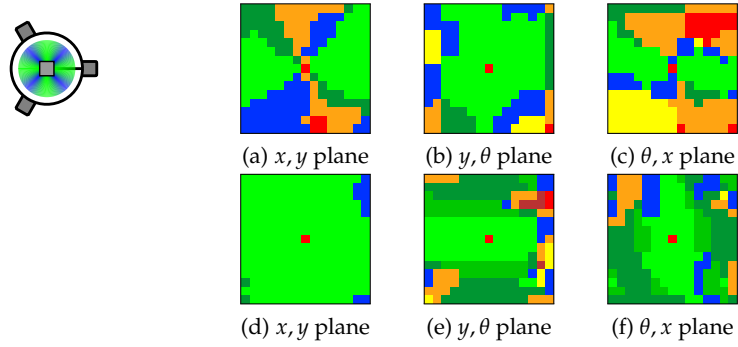


Fig. 15: Convergence results for robot with omnidirectional camera. See the caption of Fig. 14 for an explanation of the color scales.

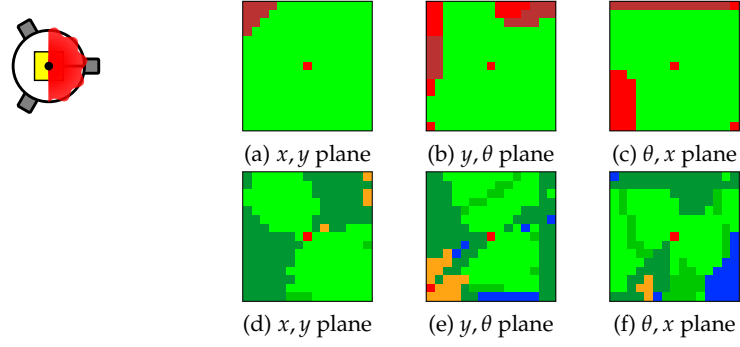


Fig. 16: Convergence results for robot with range-finder (180deg FOV). See the caption of Fig. 14 for an explanation of the color scales. For the range-finder, both control laws have large convergence radius.

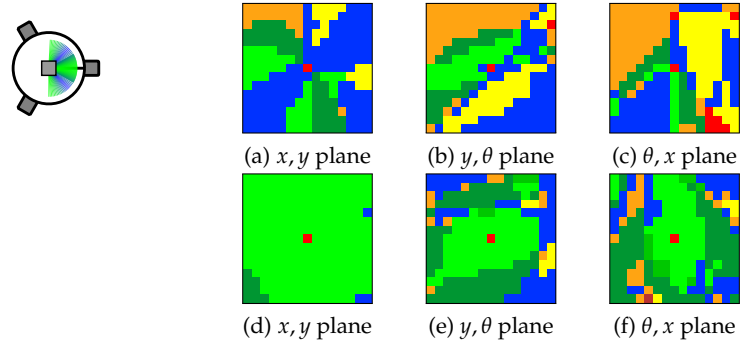


Fig. 17: Convergence results for a robot with 180deg camera.

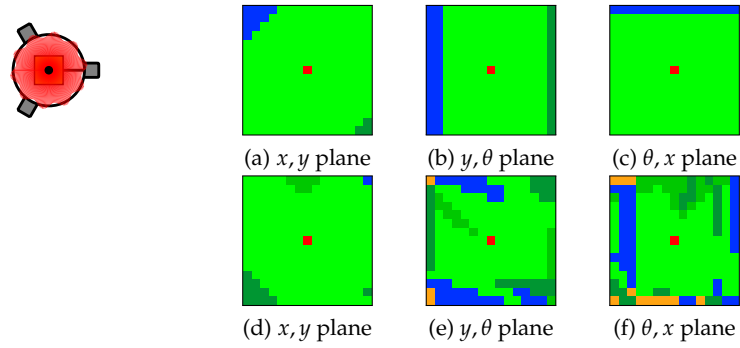


Fig. 18: Convergence results for a robot with omnidirectional range-finder.

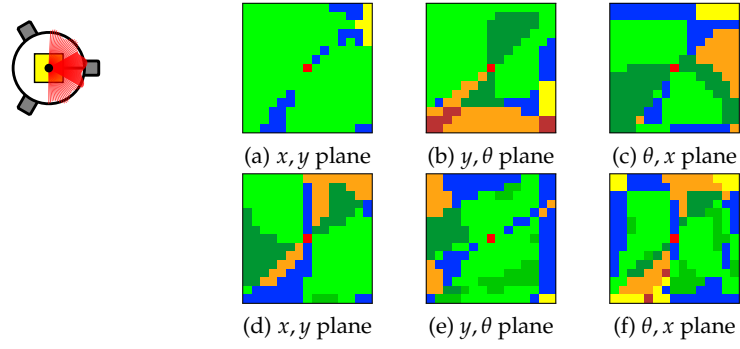


Fig. 19: Convergence results for a robot with 180deg range-finder, denser in the middle.

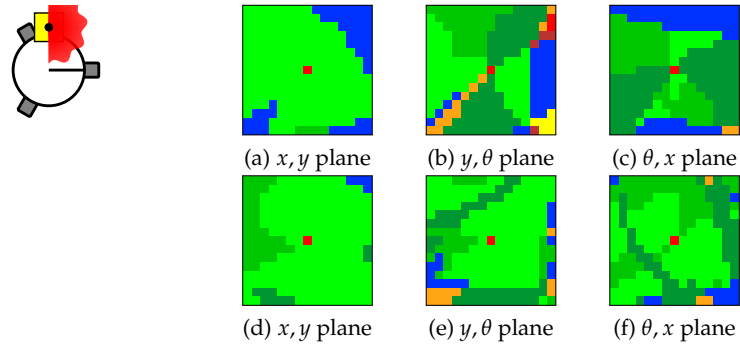


Fig. 20: Convergence results for a robot with 180deg range-finder, translated laterally.

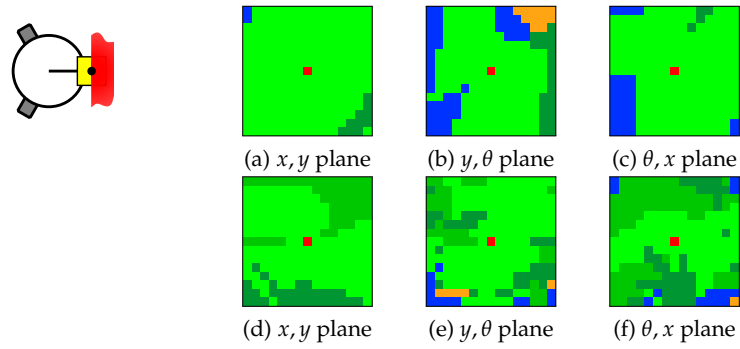


Fig. 21: Convergence results for a robot with 180deg range-finder, translated forward.

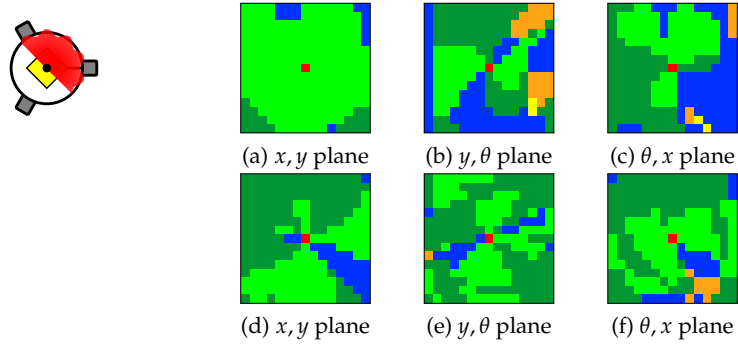


Fig. 22: Convergence results for a robot with 180deg range-finder, rotated.

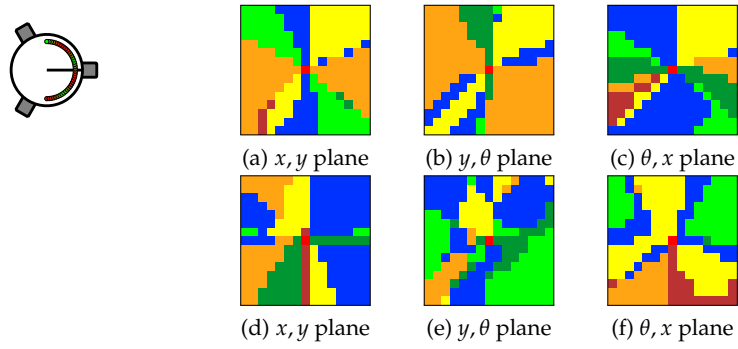


Fig. 23: Convergence results for a robot with field sampler (sensors placed on a 180deg semiring).

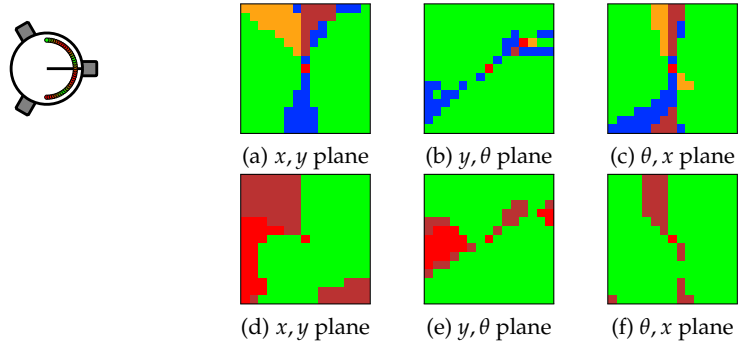


Fig. 24: Convergence results for a robot with field sampler (sensors placed on a 180deg semiring), with normalization.

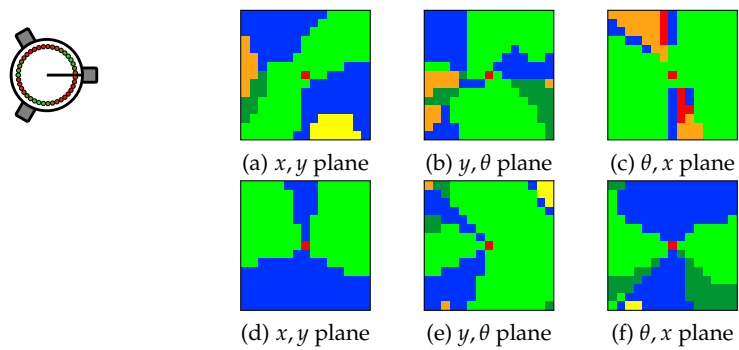


Fig. 25: Convergence results for a robot with field sampler (sensors placed on a 360deg ring), with normalization.

4 Analysis for the three canonical sensors

In the previous section, we showed by simulation that bootstrapping works for the three canonical sensors. In this section, we consider the problem of justifying this theoretically. The main results are summarized in Fig. 1. In summary, the three canonical sensors have something in common at a certain level of abstraction, as can be seen by the fact that their bootstrapped tensors are formally very similar. Because the camera and range-finder are not precisely BDS, we have to provide separate proofs of convergence. We need several preliminary results.

| | Pure BDS? | \mathcal{Y} | Sensor dynamics | Bootstrapped tensors | Convergence proof for (9) | Convergence proof for (11) |
|----------------------|-----------------------|------------------------------------|--|---|---------------------------|----------------------------|
| Field sampler | Yes. | \mathbb{R}^3 | $\dot{y}^s = \nabla_i y^s v^i + (s \times \nabla y^s)_i \omega^i$ | $T^{svi} = \nabla_j^{\mathbb{R}^3} P^{sv} Q^{ij}$ $T^{sv(i+3)} = (s \times \nabla^{\mathbb{R}^3} P^{sv})_j Q^{ij}$ | Yes. | Yes for translation. |
| Camera | No. (hidden state) | $\mathbb{R}^3 \times \mathbb{S}^2$ | $\dot{y}^s = \mu^s \nabla_i y^s v^i + (s \times \nabla y^s)_i \omega^i$ | $T^{svi} = \bar{\mu}^s \nabla_j^{\mathbb{S}^2} P^{sv} Q^{ij}$ $T^{sv(i+3)} = (s \times \nabla^{\mathbb{S}^2} P^{sv})_j Q^{ij}$ | Yes. | Yes for rotation. |
| Range-finder | No. (nonlinearity) | $\mathbb{R}^3 \times \mathbb{S}^2$ | $\dot{y}^s = (\nabla_i \log y^s - s_i^*) v^i + (s \times \nabla y^s)_i \omega^i$ | $T^{svi} = \nabla_j^{\mathbb{S}^2} \beta(P^{sv}) Q^{ij}$ $T^{sv(i+3)} = (s \times \nabla^{\mathbb{S}^2} P^{sv})_j Q^{ij}$ | Yes for rotation. | Yes for rotation. |

Fig. 1: Summary of the results in this section. The third column shows the equation for the sensor dynamics: as one can see, these three apparently different sensors have a somewhat similar dynamics. The next column shows the tensor learned; for the camera and range-finder, which are not exactly BDS, these tensors could be considered a “projection” of the nonlinear dynamics to the BDS space. The last two columns indicates whether we have a formal proof that the general control laws for BDS work for the particular sensors—see the text for details. The tensor \mathbf{P} is the covariance of \mathbf{y} ; the tensor \mathbf{Q} is the covariance of \mathbf{u} ; $\bar{\mu}^s$ is the average nearness (inverse of distance) in direction s .

We start with a series of definitions and results common for all sensors. We assume the reader to be familiar with basic Lie group theory [18]. Using the standard notation, given a group \mathcal{G} and two elements $\mathbf{a}, \mathbf{b} \in \mathcal{G}$, we indicate their product as $\mathbf{ab} \in \mathcal{G}$, and \mathbf{a}^{-1} is the inverse of \mathbf{a} .

Definition 6. Let \mathcal{C} be the *configuration space* in which the robot moves. We assume that \mathcal{C} is a subgroup of $\text{SE}(3)$.

Example. Examples of subgroups of $\text{SE}(3)$ are: $\text{SE}(3)$ itself; $\text{SE}(2)$ (planar rototranslations), $\text{SO}(3)$ (pure rotations), \mathbb{R}^3 (pure translations), or \mathbb{R} , for a robot constrained to live on a straight line⁷.

We assume that the underlying dynamics is a rigid body controlled in velocity. For $\mathcal{C} = \text{SE}(3)$ the commands \mathbf{u} are the linear and angular veloc-

⁷ The researcher that worked with it would agree this is a good model for the Pioneer 3-AT from Activmedia.

ity v, ω . Let $t \in \mathbb{R}^3$, $R \in \text{SO}(3)$ be position and attitude, and $\hat{\cdot}$ be the hat map [18]. Then the dynamics are $\dot{t} = Rv$; $\dot{R} = R\hat{\omega}$.

We already introduced the sensel space \mathcal{Y} , but it has not been given any structure yet. Here we characterize it by its interaction with \mathcal{C} .

Definition 7. Let \mathcal{Y} be the *sensel space*. We assume that it is a metric space, where for two sensel positions $s_1, s_2 \in \mathcal{Y}$, the function $d(s_1, s_2)$ indicates the distances between the positions. We also assume that there is a left action of \mathcal{C} on \mathcal{Y} . In particular, for every $q \in \mathcal{C}$ and $s \in \mathcal{Y}$, we can define the element $qs \in \mathcal{Y}$, and $q_1(q_2s) = (q_1q_2)s$.

Example. For example, for a pan-tilt-roll “robotic” camera, $\mathcal{Y} = \mathbb{S}^2$, $\mathcal{C} = \text{SO}(3)$, and the action qs corresponds to applying the rotation q to $s \in \mathbb{S}^2$.

Finally, we have to give some structure to the world around the robot. “World” is everything needed to compute the sensor output, apart from the robot pose. For a range-finder, the world includes the 3D environment structure; for a camera, it includes the texture, reflectance, and illumination information as well. We use a construction typical of stochastic geometry [14, 17]: we assume that the set of worlds \mathcal{W} can be factorized into a “shape” and “pose” component, in the sense that, for each world, there are many others that share the same shape (including color, texture, etc.), but rototranslated. Therefore, we let $\mathcal{W} \equiv \mathcal{S} \times \text{SE}(3)$, where \mathcal{S} is called *shape space*. We write an element of \mathcal{W} as a tuple $\langle s, p \rangle$, with $s \in \mathcal{S}$ and $p \in \text{SE}(3)$.

All three sensors are “relative”, in the following sense.

Definition 8. Given a sensel space \mathcal{Y} , a pose space \mathcal{C} , and a shape-pose space $\mathcal{W} \equiv \mathcal{S} \times \text{SE}(3)$, the map $y : \mathcal{W} \times \mathcal{C} \times \mathcal{Y} \rightarrow \mathcal{O}$ corresponds to a *relative sensor* if the following two properties hold for all $x \in \mathcal{C}$.

$$y(\langle s, p \rangle, q, s) = y(\langle s, xp \rangle, xq, s) \quad [\text{P1}] \quad (1)$$

$$y(\langle s, p \rangle, q, s) = y(\langle s, p \rangle, qx^{-1}, xs) \quad [\text{P2}] \quad (2)$$

Remark 9. Property P1 corresponds to the fact that there is an intrinsic ambiguity in choosing the frame of reference. The world and the robot have both a pose with respect to some fixed coordinate frame, but the output of the sensor depends only of the *relative* pose $q^{-1}p$ (let $x = q^{-1}$ in (1) to see this fact). Property P2 describes the fact that the robot is “carrying” the sensor: ultimately the output at sensel s depends only on qs , therefore it is invariant if we apply x to s and multiply q by x on the right.

Remark 10. Is this the most general case? Actually, no. There are a couple of assumptions that one would want to relax in future work. For example, we are assuming that the sensor is instantaneous, that the output depends only on the current pose and not from the history. The other assumption is that we assume that all sensors are rigidly mounted on the same rigid body. We cannot deal with articulated bodies, or even a pan-tilt camera.

The bootstrapping strategy proposed in Section 2, specifically equations (3)-(5), is described by means of statistical operators such as expectations. To predict the outcome, we have to specify something about the probability distribution of the stimuli that the agent experienced. Firstly, we give it a name.

Definition 11. Let $p_T(\langle s, p \rangle, q)$ be the *training probability distribution* of worlds/poses that the agent has experienced during its bootstrapping phase.

We want the training distribution to have some regularity. We describe the regularity with the language of Lie groups.

Definition 12. Define the set *symmetry group* of p_T as the subgroup Sym of \mathcal{C} such that, for all $x \in \text{Sym}$, $p_T(\langle s, p \rangle, q) = p_T(\langle s, xp \rangle, q)$.

Example. Consider a planar robot ($\mathcal{C} = \text{SE}(2)$), and assume we believe that, at the end of bootstrapping, the robot experience did not privilege one particular orientation over the others. Then we would set Sym to be the group of planar rotations.

At this point we are ready to give a technical definition and relative proposition, on which many other results are based.

Definition 13. Consider two couples of sensels $(s_1, v_1), (s_2, v_2)$, where $s_1, v_1, s_2, v_2 \in \mathcal{Y}$. We call the training distribution *mixing* if $d(s_1, v_1) = d(s_2, v_2)$ implies that there exists a $x \in \text{Sym}$ such that $(s_2, v_2) = (xs_1, xv_1)$.

Proposition 14. For a mixing training distribution, the expectation of any function of two sensels $s, v \in \mathcal{Y}$ is only a function of their distance; for all functions ϕ , we can write $\mathbb{E}\{\phi(y^s, y^v)\}$ as $\varphi(d(s, v))$ for some other function φ .

Proof. The proof consists in showing that $\mathbb{E}\{\phi(y^s, y^v)\}$ has the same value for two couples of sensels $(s_1, v_1), (s_2, v_2)$ that have the same distance. Write the expectation for (s_2, v_2) by showing the dependence of y on the pose q and the world $\langle s, p \rangle$.

$$\begin{aligned}
\mathbb{E}\{\phi(y^{s_2}, y^{u_2})\} &= \mathbb{E}\{\phi(y(\langle s, p \rangle, q, s_2), y(\langle s, p \rangle, q, v_2))\} \\
&\quad \text{By assumption, } s_2 = xs_1 \text{ and } u_2 = xv_1 \text{ for some } x \in \text{Sym.} \\
&= \mathbb{E}\{\phi(y(\langle s, p \rangle, q, xs_1), y(\langle s, p \rangle, q, xv_1))\} \\
&\quad \text{Because this is a relative sensor, property (2) holds.} \\
&= \mathbb{E}\{\phi(y(\langle s, p \rangle, qx, s_1), y(\langle s, p \rangle, qx, v_1))\} \\
&\quad \text{Now using property (1) applied to } (qx)^{-1}. \\
&= \mathbb{E}\{\phi(y(\langle s, (qx)^{-1}p \rangle, (qx)^{-1}qx, s_1), \\
&\quad y(\langle s, (qx)^{-1}p \rangle, (qx)^{-1}qx, v_1))\} \\
&\quad \text{Simplifying, and using the fact that } (qx)^{-1} = x^{-1}q^{-1}. \\
&= \mathbb{E}\{\phi(y(\langle s, x^{-1}q^{-1}p \rangle, e, s_1), y(\langle s, x^{-1}q^{-1}p \rangle, e, v_1))\} \\
&\quad \text{If } x \text{ is in the group of symmetries Sym, } x^{-1} \text{ is as well.} \\
&\quad \text{Using the mixing property (Definition 13), we can remove } x^{-1}. \\
&= \mathbb{E}\{\phi(y(\langle s, q^{-1}p \rangle, e, s_1), y(\langle s, q^{-1}p \rangle, e, v_1))\} \\
&\quad \text{Reusing property (1) in the other direction.} \\
&= \mathbb{E}\{\phi(y(\langle s, p \rangle, q, s_1), y(\langle s, p \rangle, q, v_1))\} \\
&= \mathbb{E}\{\phi(y^{s_1}, y^{v_1})\}
\end{aligned}$$

Because $\mathbb{E}\{\phi(y^s, y^v)\}$ has the same value for all couples of sensel with a fixed distance, it must be a function of only the distance. \square

Corollary 15. For a relative sensor in the mixing case, the covariance of two sensels is a function of only their distance: $\text{cov}(y^s, y^v) = f(d(v, s))$.

Proof. This follows directly from Proposition 14, applied to the function $\phi(y^s, y^v) = (y^s - \mathbb{E}\{y^s\})(y^v - \mathbb{E}\{y^v\})$. \square

Proposition 16. In a mixing environment, the expected value of the sensels does not depend on s : $\mathbb{E}\{y^s\} = \bar{y}$.

Proof. Apply Proposition 14 with $s = u$ and the function $\phi(y^s, y^s) = y^s$. The expectation depends on s only through $d(s, s) = 0$, and therefore it is independent of s . \square

Corollary 17. Under the conditions of Lemma 14, the gradient of y with respect to the sensor space has expected value 0: $\mathbb{E}\{\nabla y^s\} = 0$.

Proof. This is simple consequence of the linearity of the expectation: $\mathbb{E}\{\nabla y(s)\} = \nabla \mathbb{E}\{y(s)\} = \nabla \bar{y} = 0$. \square

We define a property of the environment useful in the future.

Definition 18. We call the environment *monotone* if the covariance of the values of two sensels is a monotone function of the distance between the sensels.

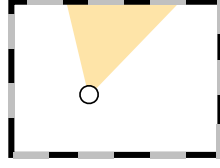


Fig. 2: Example of non monotone environment.

Remark 19. While this fact might be intuitive, it is not always true. It is true in general that the covariance reaches the maximum when the distance is 0, however, then it is not always monotonically decreasing. Usually that means that there is some structure in the environment. We offer an intuitive example in Fig. 2: suppose that a robot with a camera is in an environment that (on average) appears periodic on the retina with period $\Delta\theta$; then the correlation between pixels will be an oscillatory function of the distance with period approximately $\Delta\theta$.

4.1 Analysis of field sampler

We start with our definition of a field sampler.

Definition 20. Let the sensels space be $\mathcal{Y} = \mathbb{R}^3$. The sensor \mathbf{y} is a field sampler if there exists a field $\mathcal{H} : \mathbb{R}^3 \rightarrow \mathbb{R}$ such that $y^s = \mathcal{H}(\mathbf{t} + \mathbf{R}\mathbf{s})$, where $\mathbf{t} \in \mathbb{R}^3$ and $\mathbf{R} \in \text{SO}(3)$ are the agent position and attitude.

We can show that a field tensor is indeed a perfect BDS, and therefore all the relevant results from Section 2 apply.

Proposition 21. A field sampler is a BDS, because its observations dynamics are bilinear in \mathbf{y} and $\mathbf{u} = (\mathbf{v}, \boldsymbol{\omega})$:

$$\dot{\mathbf{y}}^s = (\nabla_i y^s) \mathbf{v}^i + (\mathbf{s} \times \nabla y^s)_i \boldsymbol{\omega}^i.$$

Proof. The proof is simple and consists of simple algebraic manipulations. Nevertheless it allows to introduce several notations and conventions. The derivative of the observations are given by:

$$\dot{\mathbf{y}}^s = \nabla \mathcal{H}|_{\mathbf{z}=\mathbf{t}+\mathbf{R}\mathbf{s}} \cdot (\mathbf{R}\mathbf{v} + \mathbf{R}\dot{\boldsymbol{\omega}}\mathbf{s}), \quad (3)$$

where we used the fact that $\dot{\mathbf{t}} = \mathbf{R}\mathbf{v}$ and $\dot{\mathbf{R}} = \mathbf{R}\dot{\boldsymbol{\omega}}$. We want to express $\nabla \mathcal{H}$ as a function of \mathbf{y} . Note that, inverting the sensor model, we obtain $\mathcal{H}(\mathbf{z}) = y(\mathbf{R}^T(\mathbf{z} - \mathbf{t}))$. Deriving that relation, we obtain $\nabla \mathcal{H} \cdot \mathbf{x} = \nabla y|_{\mathbf{s}=\mathbf{R}^T(\mathbf{z}-\mathbf{t})} \cdot \mathbf{R}^T \mathbf{x}$. Substituting in (3), we obtain:

$$\begin{aligned}
\dot{y}^s &= \nabla y|_{s=\mathbf{R}^T(z-t)} \mathbf{R}^T (\mathbf{R}v + \mathbf{R}\hat{\omega}s) \\
&\quad \text{Because } \mathbf{R}^T = \mathbf{R}^{-1} \\
&= \nabla_i y^s (v + \hat{\omega}s)^i
\end{aligned} \tag{4}$$

$$\begin{aligned}
&\quad \text{Using the property } a^T \hat{b}c = a^T (b \times c) = (b \times a)^T c. \\
&= (\nabla_i y^s) v^i + (s \times \nabla y^s)_i \omega^i.
\end{aligned} \tag{5}$$

The expression (4) that we find is bilinear in y and $u = (v, \omega)$, therefore a field sampler is a BDS. \square

We can compute the exact form for the learned tensor \mathbf{T} . Assuming the general case of a fully actuated rigid body in $\text{SE}(3)$, the tensor \mathbf{T} has 6 components for the last index. The first three ($1 \leq i \leq 3$) correspond to linear velocity, and the last three ($4 \leq i \leq 6$) to the angular velocity.

Proposition 22. *The learned tensor for a field sampler is*

$$\mathbf{T}^{svi} = \nabla_j \mathbf{P}^{sv} \mathbf{Q}^{ij}, \tag{6}$$

$$\mathbf{T}^{sv(i+3)} = (s \times \nabla \mathbf{P}^{sv})_j \mathbf{Q}^{ij}. \tag{7}$$

Proof. We show the computation for the linear velocity components:

$$\begin{aligned}
\mathbf{T}^{svi} &\triangleq \mathbb{E}\{(y^s - \bar{y}^s) \dot{y}^v v^i\} = \mathbb{E}\{(y^s - \bar{y}^s) (\nabla_j y^v) v^j v^i\} \\
&= \nabla_j \mathbb{E}\{(y^s - \bar{y}^s) y^v\} \mathbb{E}\{v^j v^i\} = \nabla_j \mathbf{P}^{sv} \mathbf{Q}^{ij}.
\end{aligned}$$

The formula for the others is obtained similarly. \square

Proposition 23. *For a mixing training distribution, the condition of Proposition 5 are satisfied for pure translation ($\mathcal{C} = \mathbb{R}^3$), and hence the simplified control (11) can be used.*

Proof. The commutation condition instantiated for the field sampler requires us to verify that we can interchange the order of ∇ and \mathbf{P} in (6). Because the environment is mixing, we know from Proposition 14 that the covariance can be written as a function of the sensel distance: $\mathbf{P}^{sv} = f(d(s, v))$. Therefore, when \mathbf{P} is used as a linear operator, it corresponds to smoothing with a radially symmetric function. Noting that in \mathbb{R}^3 gradient and radially symmetric smoothing commute concludes the proof. \square

In conclusion, a field sampler is a sensor whose dynamics is precisely that of a BDS, and therefore, the analysis is quick and compact.

4.2 Analysis of camera

In general, the sensel space of a camera is $\mathcal{Y} = \mathbb{R}^3 \times \mathbb{S}^2$: each pixel captures the light arriving to a particular focus point (in \mathbb{R}^3) from a particular di-

rection on the unit sphere (\mathbb{S}^2). For simplicity, we consider a central camera with only one focus point, so that the sensel space is just $\mathcal{Y} = \{\mathbf{0}\} \times \mathbb{S}^2$.

Proposition 24. *Let $y^s, s \in \mathbb{S}^2$, be the luminance signal captured by the camera. Let μ^s be the nearness, the inverse of the distance in direction s . Then the dynamics of \mathbf{y} are*

$$\dot{\mathbf{y}}^s = \mu^s \nabla_i y^s \mathbf{v}^i + (s \times \nabla y^s)_i \boldsymbol{\omega}^i. \quad (8)$$

See [9] for a proof. From this it follows that a camera is a BDS only for pure rotation, or with the environment at infinity, because of the dependence on the hidden state μ .

However, we can show that a useful BDS approximation can be learned. What happens is that, when learning the observation dynamics, the hidden state μ^s gets filtered out and appears only as a multiplicative factor in the BDS tensor.

Proposition 25. *Assuming that nearness and luminance are independent in p_T , the learned tensors for a camera are*

$$\begin{aligned} T^{svi} &= \bar{\mu}^s \nabla_j P^{sv} Q^{ij}, \\ T^{sv(i+3)} &= (s \times \nabla P^{sv})_j Q^{ij}. \end{aligned} \quad (9)$$

Proof. The proof is similar as Proposition 22: write the definition of T^{svi} , substitute (8) and carry on the computation. \square

Because the camera is not a BDS, we cannot use the stock results for convergence. However, the control law (9), when instantiated for the camera, has the same form as the one we studied in our previous work on bioplausible visual control [9]. Referring to those results, we can say that (9) locally converges. As for the convergence of the simplified control (11), we can prove the analogous of Proposition 23, this time for rotation rather than translation.

Proposition 26. *In a mixing environment, ignoring the conditions at the borders of the sensels area, the condition in Proposition 5 is satisfied for rotations ($\mathcal{C} = \text{SO}(3)$), and hence the simplified control law (11) can be used.*

Proof. For compactness, define the differential operator $y(s) \mapsto s \times \nabla y(s)$ as Sy . We have to prove that the application of the covariance and S commute: $SP = PS$. Just like Proposition 23, start by noticing that, because the environment is mixing, the covariance is a function of the sensel distance: $P^{sv} = f(d(s, v))$. Therefore, it acts as a convolution on the sphere. We indicate the convolution of a function y defined on the sphere with a kernel f by f_*y . An explicit expression is given by:

$$(f_*y)(s) = \int f(d(s, v))y(v)dv.$$

One can see by direct computation that f_* is self-adjoint:

$$\langle\langle \mathbf{y}_1, f_* \mathbf{y}_2 \rangle\rangle = \langle\langle f_* \mathbf{y}_1, \mathbf{y}_2 \rangle\rangle. \quad (10)$$

Let $\mathbf{r} \in \text{SO}(3)$. Define the rotated function $\mathbf{r} \circ \mathbf{y}$ as $(\mathbf{r} \circ \mathbf{y})(s) = \mathbf{y}(\mathbf{r}s)$. We can prove that rotation and convolution commute:

$$\mathbf{r} \circ (f_* \mathbf{y}) = f_* (\mathbf{r} \circ \mathbf{y}).$$

This can be seen by direct computation:

$$\begin{aligned} (\mathbf{r} \circ (f_* \mathbf{y}))(s) &= \int f(d(\mathbf{r}s, v)) \mathbf{y}(v) dv \\ &\quad \text{Change of variable: } v = \mathbf{r}u, dv = du. \\ &= \int f(d(\mathbf{r}s, \mathbf{r}u)) \mathbf{y}(\mathbf{r}u) du \\ &\quad \text{The distance is invariant to rotations.} \\ &= \int f(d(s, u)) \mathbf{y}(\mathbf{r}u) du \\ &= f_* (\mathbf{r} \circ \mathbf{y}). \end{aligned}$$

Let $\mathbf{y}_1, \mathbf{y}_2 : \mathbb{S}^2 \rightarrow \mathbb{R}$, and consider the inner product of \mathbf{y}_1 with $\mathbf{r} \circ (f_* \mathbf{y}_2)$. By the commutation property,

$$\langle\langle \mathbf{y}_1, \mathbf{r} \circ (f_* \mathbf{y}_2) \rangle\rangle = \langle\langle \mathbf{y}_1, f_* (\mathbf{r} \circ \mathbf{y}_2) \rangle\rangle.$$

Moreover, using (10), we get

$$\langle\langle \mathbf{y}_1, \mathbf{r} \circ (f_* \mathbf{y}_2) \rangle\rangle = \langle\langle f_* \mathbf{y}_1, \mathbf{r} \circ \mathbf{y}_2 \rangle\rangle.$$

The next step is computing the directional derivative of both sides with respect to \mathbf{r} at $\mathbf{r} = \text{Id}$. As explained in more detail in [9], the generic formula is

$$D_{\mathbf{r}} \mathbf{y}(\mathbf{r}s)|_{\mathbf{r}=\text{Id}} \cdot \boldsymbol{\omega} = (s \times \nabla \mathbf{y}(s))_i \omega^i.$$

This leads to $\langle\langle \mathbf{y}_1, s \times \nabla (f_* \mathbf{y}_2) \rangle\rangle = \langle\langle f_* \mathbf{y}_1, s \times \nabla \mathbf{y}_2 \rangle\rangle$, or, using a shorter notation

$$\langle\langle \mathbf{y}_1, S(f_* \mathbf{y}_2) \rangle\rangle = \langle\langle f_* \mathbf{y}_1, S \mathbf{y}_2 \rangle\rangle.$$

We can move back the convolution to the right in the second term to obtain

$$\langle\langle \mathbf{y}_1, S(f_* \mathbf{y}_2) \rangle\rangle = \langle\langle \mathbf{y}_1, f_* (S \mathbf{y}_2) \rangle\rangle.$$

Because this holds for all $\mathbf{y}_1, \mathbf{y}_2$, we have proved that $S f_* = f_* S$, which, in the case of mixing environment, is equivalent to $\mathbf{S} \mathbf{P} = \mathbf{P} \mathbf{S}$. \square

4.3 Analysis for range-finder

Each reading of a range-finder measures the distance from a an origin point (in \mathbb{R}^3) to the closest obstacle in a certain direction (in \mathbb{S}^2). Like the camera, in principle, the sensel space for a range-finder is $\mathcal{Y} = \mathbb{R}^3 \times \mathbb{S}^2$, but for simplicity we consider the case where all rays have the same origin. We start by providing an expression for the observation dynamics—even though range-finders are popular sensors, we could not find this in the published literature.

Proposition 27. *Let y^s be the range reading (distance to the obstacle in direction s). Then the dynamics of a range-finder are*

$$\dot{y}^s = (\nabla_i \log y^s - s_i^*) v^i + (s \times \nabla y^s)_i \omega^i. \quad (11)$$

Proof. (This proof is due to Shuo Han) The model for the rotation part is analogous to the field-sampler and camera. Hence we are only concerned in proving the result for translation. For clarity, we use the more widespread notation, and let the range readings be σ (rather than y). Write $\sigma = \sigma(s, t)$ as a function of the direction s and the robot position $t \in \mathbb{R}^3$. Then we have to prove that

$$\frac{\partial}{\partial t} \sigma(s, t) = \nabla \log \sigma(s, 0) - s^T.$$

Without loss of generality, we can assume we are computing the derivative at $t = 0$. In a neighbourhood of 0, it holds that

$$\|t + \sigma(s)s\| = \sigma \left(\frac{t + \sigma(s)s}{\|t + \sigma(s)s\|}, 0 \right), \quad (12)$$

as can be seen by geometric inspection of Fig. 3. The proof is based on the

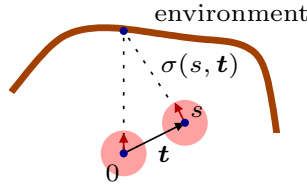


Fig. 3: Geometry of range-finder sensing.

implicit function theorem applied to the relation (12). Define the function $n(v) : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ as the vector v normalized by its module: $n(v) \triangleq v / \|v\|$. Then the following holds:

$$F(\sigma, s, \mathbf{t}) = \|\mathbf{t} + \sigma(s)s\| - \sigma(n(\mathbf{t} + \sigma(s)s), 0) = 0.$$

We can compute the derivative $\frac{\partial}{\partial \mathbf{t}} \sigma(s, \mathbf{t})$ using the implicit function theorem applied to F :

$$\frac{\partial \sigma}{\partial \mathbf{t}} = \left(\frac{\partial F}{\partial \sigma} \right)^{-1} \frac{\partial F}{\partial \mathbf{t}}.$$

To this end, we first recall that $\frac{\partial}{\partial v} \|v\| = \frac{v^T}{\|v\|}$, and we compute the derivative of $n(v)$ as

$$\begin{aligned} \frac{\partial}{\partial v} n(v) &= \frac{\partial}{\partial v} \frac{v}{\|v\|} = \frac{I}{\|v\|} + v \frac{\partial}{\partial v} \frac{1}{\|v\|} = \frac{I}{\|v\|} + v \frac{\partial}{\partial v} \frac{1}{\|v\|} = \frac{I}{\|v\|} - \frac{v}{\|v\|^2} \frac{\partial}{\partial v} \|v\| \\ &= \frac{I}{\|v\|} - \frac{v}{\|v\|^2} \frac{v^T}{\|v\|} = \frac{1}{\|v\|} \left(I - \frac{vv^T}{\|v\|^2} \right) = \frac{1}{\|v\|} \left(I - n(v)n(v)^T \right). \end{aligned}$$

We use the shortcut $x = \mathbf{t} + \sigma(s)s$, and $\sigma_0(s) = \sigma(s, 0)$.

$$\frac{\partial F}{\partial \mathbf{t}} = \frac{x^T}{\|x\|} - \nabla_u \sigma_0(u)(1 - uu^T)|_{u=n(x)} \frac{1}{\|x\|} \left(I - n(x)n(x)^T \right).$$

For the other, we simply obtain $\frac{\partial F}{\partial \sigma} = \frac{\partial F}{\partial p} s$. We compute $\frac{\partial \sigma}{\partial \mathbf{t}}$:

$$\begin{aligned} \frac{\partial \sigma}{\partial \mathbf{t}} &= - \frac{\partial F / \partial p}{\partial F / \partial \sigma} = - \frac{\frac{x^T}{\|x\|} - \nabla_u \sigma_0(u)(1 - uu^T)|_{u=n(x)} \frac{1}{\|x\|} (I - n(x)n(x)^T)}{\left(\frac{x^T}{\|x\|} - \nabla_u \sigma_0(u)(1 - uu^T)|_{u=n(x)} \frac{1}{\|x\|} (I - n(x)n(x)^T) \right) s} \\ &\quad \text{Simplifying the } \|x\|. \\ &= - \frac{x^T - \nabla_u \sigma_0(u)(1 - uu^T)|_{u=n(x)} (I - n(x)n(x)^T)}{x^T s - \nabla_u \sigma_0(u)(1 - uu^T)|_{u=n(x)} (I - n(x)n(x)^T) s} \end{aligned}$$

This expression is valid in a neighborhood of $\mathbf{t} = 0$. We now compute the limit as $\mathbf{t} \rightarrow 0$. We have

$$x \rightarrow \sigma(s)s, \quad \|x\| \rightarrow \sigma(s), \quad n(x) \rightarrow s.$$

Substituting all of these, we obtain

$$\frac{\partial \sigma}{\partial \mathbf{t}} = - \frac{\sigma(s)s^T - \nabla_s \sigma_0(s)(1 - ss^T) (I - ss^T)}{\sigma(s)s^T s - \nabla_s \sigma_0(s)(1 - ss^T) (I - ss^T) s}$$

Using the fact that $(I - ss^T)s = 0$, and $\nabla_s \sigma_0(s)(1 - ss^T) = \nabla_s \sigma_0(s)$ (the gradient is tangent to s), we simplify it to

$$\frac{\partial \sigma}{\partial \mathbf{t}} = - \frac{\sigma(s)s^T - \nabla_s \sigma_0(s)}{\sigma(s)} = \frac{\nabla_s \sigma_0(s)}{\sigma(s)} - s^T = \nabla_s \log \sigma(s) - s^T.$$

□

Note that the rotational part is exactly the same as the camera model (8), because rotation has the same effect on range and luminance data. The “ $-s_i^*$ ” term means that if the velocity v is in the direction on s , then the range decreases (the remaining nonlinear term $\nabla_i \log \sigma^s$ is less intuitive).

We can prove the following regarding the bootstrapping strategy.

Proposition 28. *If the training distribution is mixing (Definition 13) and the environment is monotone (Definition 18) the learned tensors for a range-finder are*

$$\begin{aligned} T^{svi} &= \nabla_j \beta(P^{sv}) Q^{ij}, \\ T^{sv(i+3)} &= (s \times \nabla P^{sv})_j Q^{ij}, \end{aligned} \quad (13)$$

where $\beta(P^{sv})$ is an element-wise scalar function of P^{sv} .

Proof. The proof for the rotational part is the same as the camera. As for translation, notice that a compact way to write the dynamics is $\dot{\sigma}^s = (\nabla_j \log \sigma^s - s_j^*) v^j$. Straight computation gives the following.

$$\begin{aligned} T^{sxi} &= \mathbb{E}\{(\sigma^s - \bar{\sigma}^s) \dot{\sigma}^x v^i\} \\ &\quad \text{Using the observation dynamics.} \\ &= \mathbb{E}\{(\sigma^s - \bar{\sigma}^s) (\nabla_j \log \sigma^x - x_j^*) v^j v^i\} \\ &\quad \text{Separating the two terms} \\ &= Q^{ij} \left[\mathbb{E}\{(\sigma^s - \bar{\sigma}^s) \nabla_j \log \sigma^x\} - \mathbb{E}\{x_j^* (\sigma^s - \bar{\sigma}^s)\} \right] \\ &\quad \text{The second term disappears given that } x_j^* \text{ is a constant.} \\ &= Q^{ij} \left[\mathbb{E}\{(\sigma^s - \bar{\sigma}^s) \nabla_j \log \sigma^x\} \right] \\ &\quad \text{We can pull out } \nabla \text{ due to linearity.} \\ &= Q^{ij} \nabla_j \mathbb{E}\{(\sigma^s - \bar{\sigma}^s) \log \sigma^x\}. \end{aligned}$$

At this point, note that if we had σ^x instead of $\log \sigma^x$ inside the expectation, the result would be P^{sx} , the covariance of σ . Define $R^{sx} = \mathbb{E}\{(\sigma^s - \bar{\sigma}^s) \log \sigma^x\}$. Then we can invoke Proposition 14 to say that R^{sx} is a function of $d(s, x)$. Because the environment is monotone, we know there is a 1-to-1 correspondence between P^{sx} and $d(s, x)$, therefore we can write R^{sx} as a function of P^{sx} . □

Because range-finders and cameras are equivalent under pure rotations, it is immediate to show convergence of (9) in that case. In particular, the equivalent of Proposition 26 holds. It is instead challenging to prove convergence of (9) for translation. The main reason is that P^{-1} does not cancel the term $\beta(P)$ in (13), due to the nonlinearity of β .

5 Conclusions and future work

This paper presents a contribution to the vast problem of bootstrapping, which consists in estimating and using models of a sensorimotor cascade, starting from uninterpreted commands and observations. We have shown that the abstraction of bilinear dynamics sensors (BDS) is general yet powerful enough to represent the main phenomena of a representative selection of robotics sensors (field samplers, cameras, and range-finders).

To the best of our knowledge, this is the first presentation of a bootstrapping agent that can provably learn to use a variety of sensors to solve the same cross-modality task. The algorithm is also simple, consisting only of a few lines of code; it is fast, being extremely parallelizable. With respect to the approach of Kuipers and his group, we focused on a more “continuous” rather than a “symbolic” solution, and this made it possible to actually prove strong results concerning the convergence of the learning process and the control law.

So far we have been focusing more on the sensors rather than actuators, as we only considered the case of fully-actuated velocity control. Previous work [9] suggests that control in velocity can be extended to control in forces/torques with relatively little effort. Instead, it seems more challenging learning a hidden state and its dynamics. This would be useful in the case of a camera, where the observation dynamics depends on the nearness. The other challenge is learning a more nonlinear model (perhaps by learning additional levels of Taylor approximations after the bilinear terms); this would be useful in the case of the range-finder, which is not a pure BDS for its nonlinearity.

Acknowledgements. Thanks to Shuo Han and Scott Livingston for precious comments on the first version of this paper.

References

1. M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida. Cognitive developmental robotics: A survey. *IEEE Trans. on Autonomous Mental Development*, 1(1):12–34, 2009.
2. Roger W. Brockett. Asymptotic stability and feedback stabilization. In R. S. Millman R. W. Brockett and H. J. Sussmann, editors, *Differential Geometric Control Theory*, pages 181–191. Birkhauser, Boston, 1983.
3. Ceriani and *et al.* Rawseeds ground truth collection systems for indoor self-localization and mapping. *Auton. Robots*, 27(4):353–371, 2009.
4. Cohen and *et al.* Functional relevance of cross-modal plasticity in blind humans. *Nature*, 389(6647):180–183, 1997.
5. Scott C. Douglas and Andrzej Cichocki. Neural networks for blind decorrelation of signals. *IEEE Trans. on Signal Processing*, 45(11):2829–2842, 1997.
6. Dileep George and Jeff Hawkins. Towards a mathematical theory of cortical micro-circuits. *PLoS Comput Biol*, 5(10):e1000532, 10 2009.

7. Etienne Grossmann, José Antonio Gaspar, and Francesco Orabona. Discrete camera calibration from pixel streams. *Computer Vision and Image Understanding*, 114(2):198 – 209, 2010.
8. J.-S. Gutmann, G. Brissson, E. Eade, P. Fong, and M. Munich. Vector field slam. In *Proceedings of the IEEE Conference on Robotics and Automation*, 2010.
9. Shuo Han, Andrea Censi, Andrew D. Straw, and Richard M. Murray. A bio-plausible design for visual pose stabilization. Technical Report CaltechCDSTR:2010.001, California Institute of Technology, 2010. (a reduced version to appear in IROS’10).
10. Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2004.
11. A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, 2001.
12. B. Kuipers. The Spatial Semantic Hierarchy. *Artificial Intelligence*, 119(1-2), 2000.
13. Benjamin Kuipers. An intellectual history of the Spatial Semantic Hierarchy. *Robotics and cognitive approaches to spatial mapping*, 38:243–264, 2008.
14. Huiling Le and David G. Kendall. The Riemannian structure of Euclidean shape spaces: A novel environment for statistics. *Annals of Statistics*, 21(3):1225–1271, 1993.
15. Thomas Lochmatter and Alcherio Martinoli. Theoretical analysis of three bio-inspired plume tracking algorithms. In *Proceedings of the IEEE Conference on Robotics and Automation*, 2009.
16. Max Lungarella, Giorgio Metta, Rolf Pfeifer, and Giulio Sandini. Developmental robotics: a survey. *Connection Science*, 15:151–190, 2003.
17. Peter W. Michor and David Mumford. Riemannian geometries on spaces of plane curves. *J. of the European Math. Soc.*, 8:1–48, 2006.
18. Shankar Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer-Verlag, Berlin, 1999.
19. Jeremy Stober, Lewis Fishgold, and Benjamin Kuipers. Learning the sensorimotor structure of the foveated retina. In *Proceedings of the Ninth International Conference on Epigenetic Robotics People*, 2009.
20. Jeremy Stober, Lewis Fishgold, and Benjamin Kuipers. Sensor map discovery for developing robots. In *AAAI Fall Symposia Series: Manifold Learning and Its Applications*, 2009.